NIST Special Publication 500-185

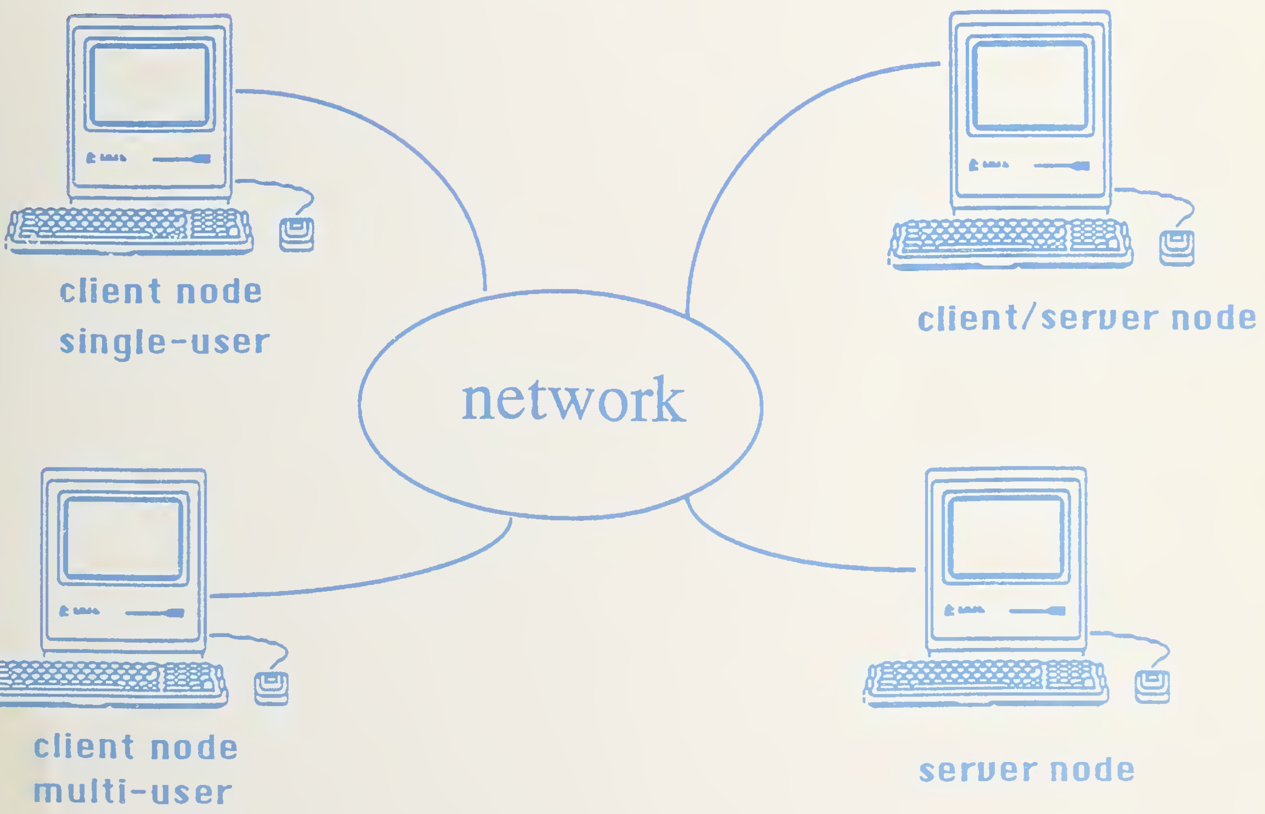
Computer Systems Technology

U.S. DEPARTMENT OF COMMERCE
National Institute of Standards and Technology

NIST

Guide to Design, Implementation and Management of Distributed Databases

Elizabeth N. Fong
Charles L. Sheppard
Kathryn A. Harvill



NIST Special Publication 500-185

Guide to Design, Implementation and Management of Distributed Databases

Elizabeth N. Fong
Charles L. Sheppard
Kathryn A. Harvill

Computer Systems Laboratory
National Institute of Standards and Technology
Gaithersburg, MD 20899

February 1991

U.S. DEPARTMENT OF COMMERCE
Robert A. Mosbacher, Secretary
NATIONAL INSTITUTE OF STANDARDS AND TECHNOLOGY
John W. Lyons, Director

Reports on Computer Systems Technology

The National Institute of Standards and Technology (NIST) has a unique responsibility for computer systems technology within the Federal government. NIST's Computer Systems Laboratory (CSL) develops standards and guidelines, provides technical assistance, and conducts research for computers and related telecommunications systems to achieve more effective utilization of Federal information technology resources. CSL's responsibilities include development of technical, management, physical, and administrative standards and guidelines for the cost-effective security and privacy of sensitive unclassified information processed in Federal computers. CSL assists agencies in developing security plans and in improving computer security awareness training. This Special Publication 500 series reports CSL research and guidelines to Federal agencies as well as to organizations in industry, government, and academia.

National Institute of Standards and Technology Special Publication 500-185
Natl. Inst. Stand. Technol. Spec. Publ. 500-185, 59 pages (Feb. 1991)
CODEN: NSPUE2

U.S. GOVERNMENT PRINTING OFFICE
WASHINGTON: 1991

For sale by the Superintendent of Documents, U.S. Government Printing Office, Washington, DC 20402

TABLE OF CONTENTS

ABSTRACT

For an organization to operate in a distributed database environment, there are two related but distinct tasks that must be accomplished. First, the distributed database environment must be established. Then, a distributed database application can be designed and installed within the environment. This guide describes both of these activities based on a development life-cycle phase framework. This guide provides practical information and identifies skills needed for systems designers, application developers, database and data administrators who are interested in the effective planning, design, installation, and support for a distributed database environment. In addition, this guide instructs system analysts and application developers with a step-by-step procedure for the design, implementation and management of a distributed DBMS application.

This guide also notes that truly heterogeneous distributed database technology is still a research consideration. Commercial products are making progress in this direction, but usually with many update and concurrency restrictions and often with severe performance penalties.

The scope of this guide is based on experiences gained in the development of a distributed DBMS environment using two off-the-shelf homogeneous distributed database management systems. In support of the successful installation of the distributed database environment, a demonstration distributed application was designed and installed.

Keywords: databases; DDBMS; distributed application; distributed database management systems; distributed environment; life-cycle phases.

ACKNOWLEDGMENTS

The technical work for this report was done through hands-on experimentation with two distributed database management systems. These distributed database management systems were provided to us by the system vendors as part of cooperative projects with the Computer Systems Laboratory (CSL). The cooperative projects were performed in the NIST/CSL Database Laboratory.

We acknowledge the contributions of Joseph Collica, and Bruce Rosen of NIST who assisted us with careful review of this paper and suggested important improvements.

1.0 INTRODUCTION

This guide provides practical information and identifies skills needed for systems designers, application developers, data and database administrators, who are interested in the effective planning, design, installation, and support for a distributed database management system (DDBMS) environment. In addition, this guide provides system analysts and application developers with a step-by-step procedure for the design, implementation and management of a DDBMS application. A distributed DBMS application is a software system which runs in the distributed database environment.

The procedure is based on a development life-cycle phase framework. At each phase in the life cycle, this guide describes the nature of the tasks, who should do these tasks, and some general guidance on accomplishing each task.

1.1 The Promises and Realities of Distributed DBMS

A distributed database environment consists of a collection of sites or nodes, connected together by a communication network. Each node has its own hardware, central processor and software which may, or may not, include a database management system (DBMS).

In the ultimate distributed database computing environment, a user will be able to access data residing anywhere in the computer network, without regard to differences among computers, operating systems, data manipulation languages, or file structures [CERI84]. Data that is actually distributed across multiple remote computers will appear to the user as if it resided on the user's own computer. This scenario is functionally limited using today's distributed database technology. True distributed database technology is still a research consideration. The functional limitations are generally in the following areas:

- o transaction management,
- o standard protocols for establishing a remote connection, and
- o independence of network topology.

Transaction management capabilities are essential to maintaining reliable and accurate databases. In some cases, today's distributed database software places the responsibility of managing transactions on the application program. In other cases, transactions are committed or rolled-back at each location independently which means that it is not possible to create a single distributed transaction. For example, multiple site updates require multiple transactions.

In today's distributed database technology, different gateway software has to be used and installed to connect nodes using different DDBMS software. Thus, connectivity among heterogeneous DDBMS nodes is not readily available (i.e., available only through selected vendor markets).

In some instances, DDBMS software is tied to a single network operating system. This limits the design alternatives for the DDBMS environment to the products of a single vendor. It is advisable to select a product that supports more than one network operating system. This will increase the possibility of successfully integrating the DDBMS software into existing computer environments.

In reality, distributed databases encompass a wide spectrum of possibilities including:

- o Remote terminal access to centralized DBMS (e.g., airline reservation system),
- o Remote terminal access to different DBMSs, but one at a time (e.g., PRODIGY, COMPUSERVE, DOW JONES, etc.),
- o Simple pairwise interconnection with data sharing that requires users to know the data location, the data access language, and the logon procedure to remote DBMS.
- o Distributed database management with a generic data definition language and a data manipulation language at all nodes.
- o Distributed update and transaction management.
- o Distributed databases with replication that support vertical and horizontal fragmentation.
- o "True" distributed database management system with heterogeneous hardware, software, and communications.

The definition of DDBMS lies anywhere along this spectrum. For the purpose of this report, the remote terminal access to data as discussed in the first two bullets above is not considered as a DDBMS because a node in the DDBMS has to have its own hardware, central processor and software.

1.2 Motivation

Some of the problems that currently frustrate managers and technicians who might otherwise be interested in exploring distributed database solutions include:

- A distributed database environment will have all of the problems associated with the single centralized database environment, but at a more complex level.

- The lack of basic step-by-step guides covering the analysis, design, and implementation towards a distributed database environment.

As discussed in [FONG88], there are many benefits offered by a distributed database management system (DDBMS). However, there are also many architectural choices which make the application design for distributed databases very complex. For an effective and productive distributed database environment, it is essential that the distributed environment be properly designed to support the expected distributed database applications. Additionally, an effective design will depend on the limitations of the DDBMS software. Thus, implementing today's distributed database technology requires identifying the functional limitations of a selected commercial product. Identification of these limitations are critical to the successful operations of an application in a distributed database environment.

1.3 Purpose and Scope

For an organization to operate in a distributed database environment, there are two related but distinct tasks that must be accomplished. First, the distributed database environment must be established. Then, a distributed database application can be designed and installed within the environment. This guide describes both of these activities.

This guide is based on the NIST Special Publication 500-154 "Guide to Distributed Database Management" [FONG88] that was produced to assist managers in both evaluating distributed database management technology for their individual environments, and in planning for an orderly migration path into a distributed database environment. The NIST Special Publication 500-154 report also outlines the distributed DBMS development life cycle phases.

The purpose of this guide is to provide organizational and technical guidance in the total life cycle development phases of:

(1) the distributed database environment, and

(2) distributed database applications.

The scope of this guide is the total development life-cycle from planning and design through implementation and support. The level of detail for each phase is very concise and is limited to brief discussions of issues and summaries of tasks.

Since each of the different life-cycle phases is often performed by different persons, the guidance at each development phase is targeted towards a different audience. For example, the guidance for the corporate strategy planning phase is given at the high management level, while the guidance for the installation of the distributed database environment is targeted at technical engineers. It is the aim of this guide to cover the total broad range of scope. Those who are not interested in the technical engineering details may skip the sections on installation and implementation.

1.4 Organization of the Guide

This guide is organized according to the life-cycle development phases for a distributed database system. The actual correspondence between the sections of this guide with the development phases is shown in figure 1.1.

- o Section 1 introduces the guide, its motivation, purpose and scope.
- o Section 2 presents a life-cycle framework for the development of a distributed database environment and a distributed database application.
- o Sections 3 and 4 describe design issues and present a brief summary of the tasks to be performed for the planning and high-level design for a distributed environment that will incorporate the requirements for the distributed database application.
- o Sections 5 and 6 describe the issues and the tasks involved during the design and installation phases of the establishment of a distributed database environment.
- o Sections 7 and 8 describe the issues and the tasks involved during the design and implementation phases of a distributed database application.
- o Section 9 describes the issues and the tasks for the support and maintenance of the environment and the applications within the environment.
- o Section 10 is the concluding remarks.

Finally, two appendices are included: Appendix A describes the sample distributed database environment as installed in the CSL Database Laboratory. Appendix B contains samples of the schema description for the demonstration distributed database application. Also contained in Appendix B are several sample distributed queries.

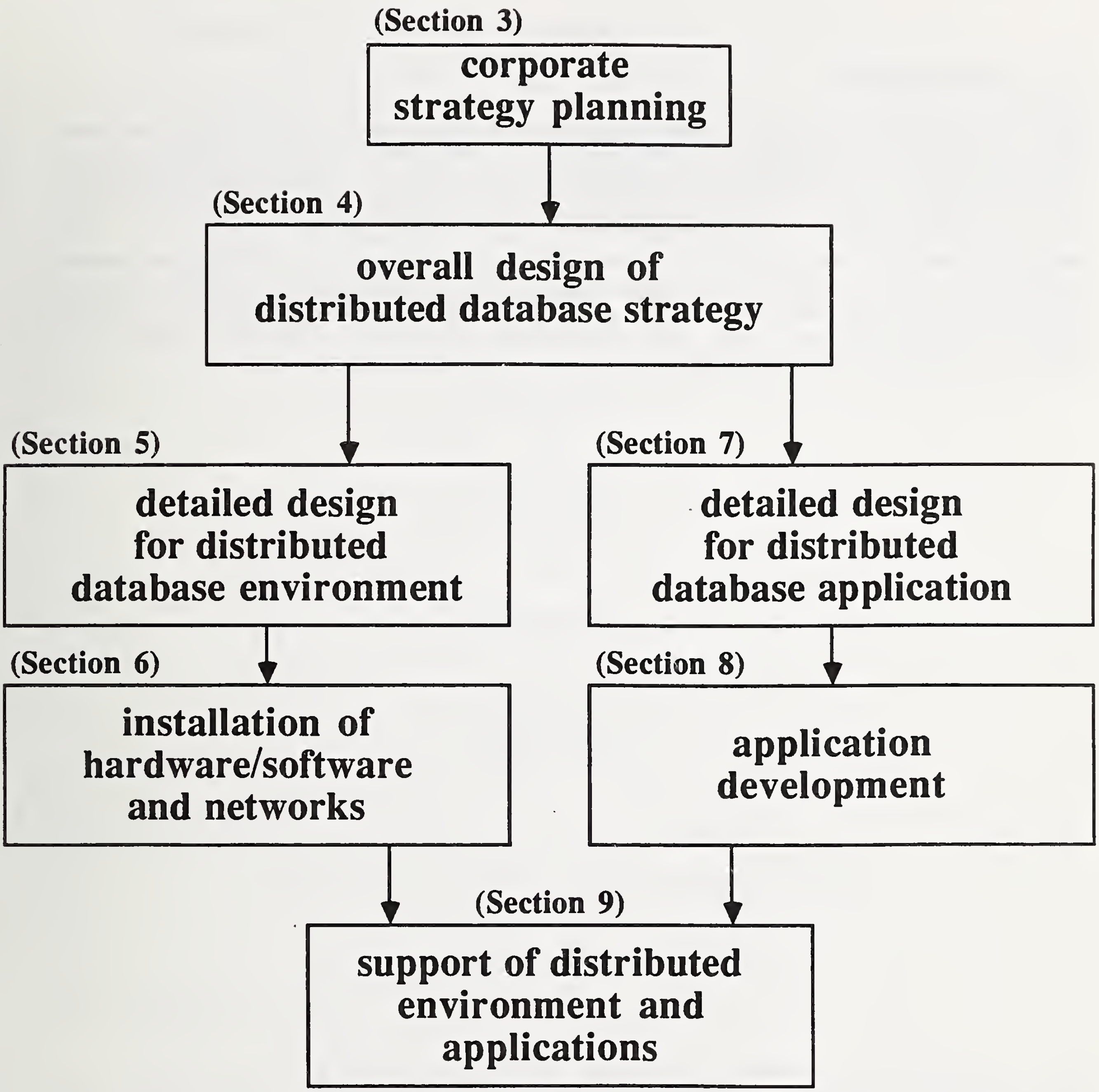

(Section 3)
corporate strategy planning
(Section 4)
overall design of distributed database strategy
(Section 5)
detailed design for distributed database environment
(Section 7)
detailed design for distributed database application
(Section 6)
installation of hardware/software and networks
(Section 8)
application development
(Section 9)
support of distributed environment and applications

Figure 1.1. Organization of the Guide by Development Phase.

1.5 Disclaimer

The discussions covering the two development cycles are presented at a generic level generalized from experiences. Most of the guidance provided is derived from experiences gained in the development of distributed DBMS environments using two off-the-shelf homogeneous distributed database management systems. In support of the successful installation of the distributed database environment, a prototype distributed application was designed and installed. The use of these products in the preparation of this guide, does not imply any recommendation or endorsement by NIST for the products or the companies.

2.0 DEVELOPMENT PHASES TO THE DISTRIBUTED DATABASE

Good corporation-wide distributed database processing is not going to happen overnight. It requires a carefully planned infrastructure within which an orderly evolution can occur. NIST Special Publication 500-154 [FONG88] describes the four major development phases: planning, designing, installing, and supporting.

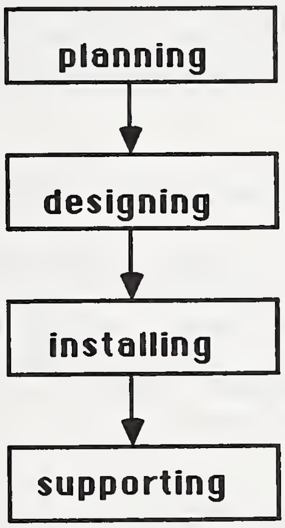


Figure 2.1 - Major Development Phases

2.1 Planning Phase

Phase 1 of figure 2.1, the planning phase, consists of the very high level management strategy planning. During the planning phase an organization must consider whether it is advantageous to migrate into a distributed environment. This guide assumes that a migration to a distributed environment is desirable and feasible, and the corporate strategy planning issues and tasks have been identified. The result of this phase is the total management's commitment for cost, resources, and a careful migration path towards a distributed database environment.

2.2 Design Phase

Phase 2 of figure 2.1, the design phase, consists of the overall design of the distributed database strategy. The overall design task involves the selection of a distributed DBMS environment in terms of the hardware, software, and the communication network for each node and how they are to be interconnected. The design of the distributed database environment must incorporate the requirements for the actual distributed database application. The overall design will divide into two main tasks: the detailed design of the distributed database environment and the detailed design of the initial distributed database application. In certain cases, the initial application may be a prototype that is intended to pave the way for the full production distributed database application.

2.3 Installation and Implementation Phase

Phase 3 of figure 2.1, the installation and implementation phase, consists of the installation and implementation of the environment which provides basic software support for the distributed DBMS application. The task of development of the distributed database application could occur in parallel with the installation of the environment.

2.4 Support and Maintenance Phase

Phase 4 of figure 2.1, the support and maintenance phase, consists of support for the distributed DBMS environment and the support and maintenance of the application. Although these support and maintenance tasks can be performed by the same people, the nature of the tasks and responsibilities are quite distinct. For example, the distributed application may require modification of report formats while the distributed environment may require modifcation to add more memory.

2.5 Detail Development Phase

Figure 1.1, as previously shown, is a more detailed view of the development phases required in establishing a distributed database environment and the development of an application. The design and installation phases for the distributed environment and the distributed application can be incorporated through two parallel tracks.

The remaining sections of this guide describe each development phase in detail. Wherever possible, descriptions of design alternatives and issues are presented. The actual development tasks are summarized in a box. Each box contains the following items:

- o a summary of tasks for the subject development step;
- o the results of the subject development step;
- o the team members who are most appropriate for performing the subject development step;
- o and some general guidelines and advice for the subject development step.

3.0 CORPORATION STRATEGY PLANNING

The main task during the strategic planning phase is to obtain the commitment of top management. The measure of this commitment is in terms of the amount of resources (both personnel and equipment) necessary for the development of a DDBMS.

The factors that must be considered during the strategy planning phase are as follows:

- o What will be the objectives of the organization's next 5-year plan?
- o How will technological changes affect the organization's way of doing business?
- o What resources are needed to plan for the development of, and migration to, a distributed database management system?
- o What tools or methods can be employed to develop and implement the plan?
- o How will outcomes be measured relative to the impact on the organization's state?

The corporate strategy plan must include detailed specifications of the total system life cycle. It must also have a realistic time table of schedules and milestones. Important consideration must be paid to the allocation of cost for new acquisitions, training of personnel, physical space requirements, and other tangible items.

During the strategic planning phase, information must be gathered on the organization's business functions and goals, related constraints and problem areas, and the organization's user groups. Only after the needed information has been gathered is it possible to develop high-level information categories and their interrelationships.

The process of developing the distributed database plan is very iterative. These activities are often performed by data administrators or information resource managers. While these individuals often have the vision to recognize the long term benefit of a DDBMS environment to an organization, they must rely on the participation and input of those in the organization who are directly involved with the business functions who use information to make decisions and manage operations. There must be considerable interaction among many different people in the organization, each of whom provides feedback in order to validate and refine the plans.

3.1 Summary of Tasks

Strategic planning must first provide a sufficient justification for the expenditure of resources necessary to migrate to a distributed environment. Only after this justification is accepted and fully approved by upper management can the task of initiating projects to design, develop, and implement a DDBMS environment and applications start.

CORPORATION STRATEGY PLANNING

Summary of Tasks:

- o Obtain top management commitment for the migration into the distributed environment.
- o Develop a detailed specification of DDBMS life cycle plan, including a time-table of schedules and mile stones.
- o Evaluate various policies for the development of site responsibilities, including considerations of data sharing and control procedures.

Results:

- o Specification of DDBMS life cycle plan.
- o Estimate of cost allocation, human resources and time span required.

Team Members;

- o Supervisory managers and data administrators.

Guidelines:

- o It is important to involve different groups of people within the organization in planning for DDBMS.
- o Establish realistic expectations for the projects.

4.0 OVERALL DESIGN OF DISTRIBUTED DATABASE STRATEGY

A distributed database environment consists of a collection of sites or nodes, connected together by a communication network. Each node has its own hardware, central processor and software which may, or may not, include a database management system (DBMS). The primary objective of a distributed DBMS is to give interactive query users and application programs access to remote data as well as local data.

4.1 Types of Distributed Environment

Individual nodes within the distributed environment can have different computing requirements. Accordingly, these nodes may have different hardware, different software, and they may be connected in many different ways. The characteristics of a DDBMS are discussed in [FONG88] and will not be repeated here. However, some of the variations possible in the distributed database environment are discussed here.

Client-Server Computing

The most basic distributed capability is remote database access from single users at a node [FINK89]. A node may be a mainframe, a minicomputer, or a microcomputer (personal computer). The node which makes the database access request is referred to as a client node, and the node which responds to the request and provides database services is referred to as the server node. The association is limited to the two parties involved - the client and the server.

Figure 4.1 provides a pictorial representation of several different configurations available under a client-server computing environment. The following are descriptions of the different configurations shown in figure 4.1:

o **Client Single User Node**

The operating environment of an individual node can be single or multi-user depending upon the operating system of that node. In a single user operating environment, a node can only be a client. Such a node may or may not have databases. For non-database client nodes, the software typically consists of front-end application programs used to access remote database server nodes. This front-end software is generally in the form of end-user interface tools such as a query language processor, a form processor, or some other application specific program written in a third generation language (i.e., C, Fortran, or Cobol). The front-end software formulates and issues user requests. It processes user requests through its

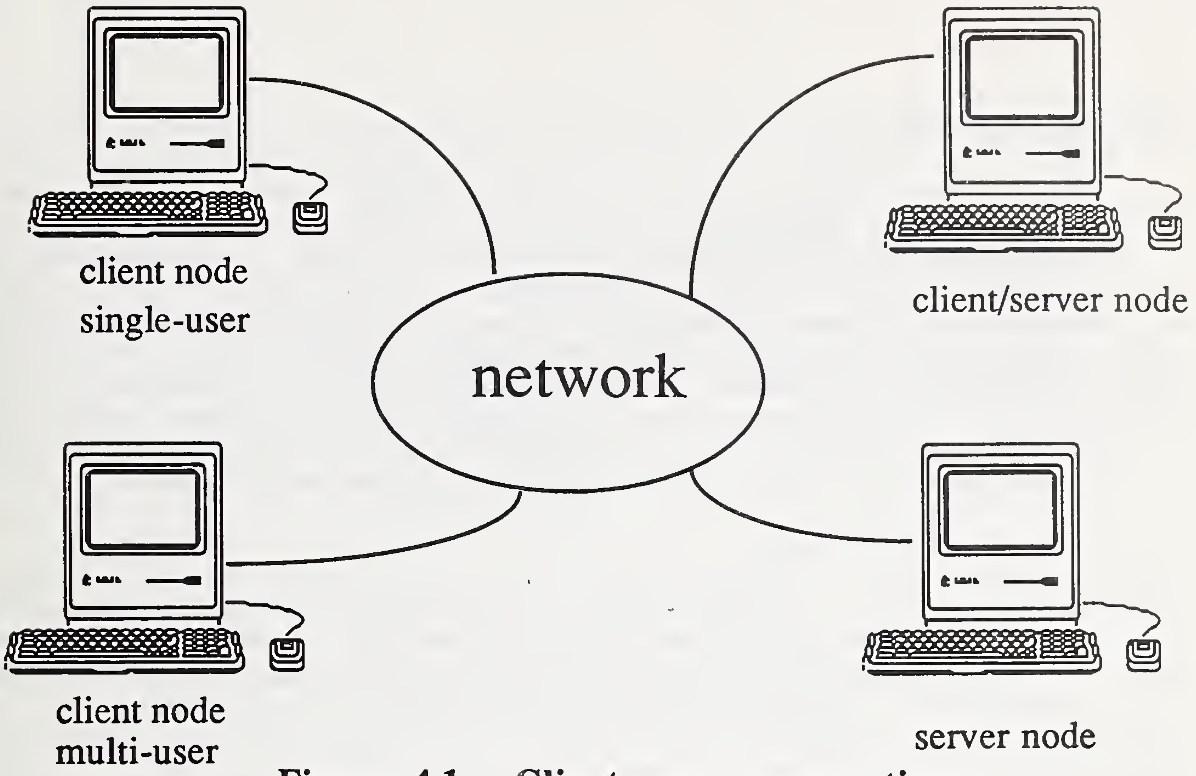


Figure 4.1. Client server computing.

established linkage with appropriate communication software. Thus, the front-end software only captures a user's request and uses communication software to send that request to a remote database node requesting its database management system to process the request. In addition to the capabilities outlined, single user nodes with databases allow local data to be included in the same query operations specified for remote data. Thus, operationally, the query results will appear as if all data is coming from a single central database.

o **Client Multi-User Node**

The functional capabilities outlined for the client single user node are expanded in the client multi-user node. This is due to the presence of a multi-user operating system at the user node. Such a configuration generally has several user processes running at the same time. At peak usage time, the presence of several user processes can cause slower response time than is achievable in a client single user node. However, the client multi-user node is more cost effective since it can allow multiple remote database accesses at different sites by different users at the same time. This is made possible through an identifiable list of remote server node locations. Additionally, as with the client single user node, the client multi-user node can include local database accesses in conjunction with accessing remote databases.

- **Server Node**

  The server node is capable of providing database services to other client requests as well as to itself. It is a special multi-user node that is dedicated to servicing remote database requests and any local processes. This means that incoming requests are serviced, but it does not originate requests to other server nodes. The functional capabilities of a server node are as follows: (1) this node must be included in the server list of some remote client node, (2) there must be an operating DBMS, and (3) there must be a continuously running process that listens for incoming database requests.

- **Client/Server Node**

  A node with a database can be a client as well as a server. This means that this node can service remote database requests as well as originate database requests to other server nodes. Thus, the client/server node can play a dual role.

Homogeneous Distributed DBMS Environment

A completely homogeneous DDBMS environment would exist when all the nodes in the distributed environment have the same DBMS, but not necessarily the same hardware and operating system. However, the communication software for each node must use the same protocol in order to send/receive requests and data.

Design and implementation of a homogeneous DDBMS environment need only involve a single vendor. Any database request issued at a client node does not need to be translated since the database language and data model are the same across all nodes in the network.

Heterogeneous Distributed DBMS Environment

In a truly heterogeneous DDBMS environment, the hardware, operating systems, communication systems and DBMSs can all be different. Different DBMSs may mean different data models along with different database languages for definition and manipulation. Any database request issued at a client node would have to be translated so that the server node responding to the request would understand how to execute the request.

There can be various degrees of heterogeneity. For example, within the distributed environment, different DBMSs can still be compatible if these different DBMSs all support the relational data model and understand SQL, a relational query language, which is an ANSI and ISO standard [FIPS127]. However, presently, even among SQL conforming systems, there is no general communication software that will accept generic SQL statements from any other SQL conforming DBMS. This is an area in which the pending Remote Data Access (RDA) standards are needed.

4.2 Selecting an Architecture for a Distributed Environment

The design of a distributed database environment can be evolutionary - by incremental interconnection of existing systems, or by developing a totally new distributed DBMS environment using the bottom-up approach. Some of the design issues in adopting either approach are described below.

Interconnection of Existing Systems

Not all organizations have the luxury of developing the distributed database environment from scratch. Already existing database management applications are costly investments which are not likely to be replaced all at once by new distributed systems. The existing environment, including hardware, software, and databases, can be preserved by providing a mechanism for producing federated systems, that is, systems comprised of autonomous software components [HEIM85], [HEIL89].

The federated approach is a practical, first-step solution, towards a distributed database environment. It accommodates a legacy of existing systems while extending to incorporate new nodes. Thus, it is very important to select DDBMS software that supports existing computing hardware and allows for expansion. Within a federated system, pairs of nodes can be coupled in ways that range from very loose, where each node is autonomous, to very tight, where each node interacts directly with the other. The various forms of coupling affect the design, execution, and functionality of the distributed applications.

The mode of coupling affects the number of translations required to exchange information between each site. Zero translations are needed when both components use the same representations. Some systems may choose to translate directly the data produced by one site to the format required by the other site. However, a more commonly used method is to translate the data first into a neutral format, and then from the neutral format translate into the target format.

o **Loose Coupling**

Loosely coupled systems are the most modular and in some ways are easier to maintain. This is because changes to the implementation of a site's system characteristics and its DBMS are not as likely to affect other sites. The disadvantage of loosely coupled systems is that users must have some knowledge of each site's characteristics in order to execute requests. Since there is very little central authority to control consistency, there is no guarantee of correctness. Furthermore, loosely coupled systems typically involve more translations which may cause performance problems.

- **Tight Coupling**

  Tightly coupled systems behave more like a single, integrated system. Users need not be aware of the characteristics of the sites fulfilling a request. With centralized control, the tightly coupled systems are more consistent in their use of resources and in their management of shared data. The disadvantage of tight coupling is that because sites are inter-dependent, changes to one site are likely to affect other sites. Also, some sites may object to the loss of freedom to the central control mechanisms necessary to maintain the tight coupling of all the systems.

Cooperation Between Sites

For a truly distributed DBMS environment, there can be a variety of ways to specify "cooperation between sites." One way of classifying the distributed environment is to define the amount of transparency offered to the users. Another way is to define the amount of site autonomy available to each site, and their cooperative process with each other.

- **Degrees of Transparency**

  Transparency is the degree to which a service is offered by the DDBMS so that the user does not need to be aware of it. One example of transparency is location transparency which means users can retrieve data from any site without having to know where the data is located.

- **Types of Site Autonomy**

  Site autonomy refers to the amount of independence that a site has in making policy decisions. Some examples of policy decisions include ownership of data, policies for accessing the data, policies for hours and days of operation, and human support. Additionally, all modifications to the site's data structures need to be approved by the cooperating federation of data administrators.

Interconnection of Newly Purchased Systems

An organization will have a lot more freedom if it decides to establish a distributed database environment from scratch. Currently, in the marketplace, vendors are offering homogeneous DDBMS with a compatible family of software. How to configure the hardware, software, and communications equipment will be discussed in section 5. However, this approach can lock the organization into a single vendor's proprietary distributed database products.

Other approaches in selecting distributed architecture choices are as follows:

- o Identical DBMS products at each node, with possibly different hardware environments, but a single proprietary communications network to interconnect all sites.
- o Standard conforming DBMS products at each node that rely on standard communications protocols. Considerations for standard products are discussed in section 4.3.
- o Different DBMSs, using the same data model (e.g., relational), interconnected by a single or standard communications protocol.
- o Different DBMSs, using different data models (e.g., relational, object-oriented, etc), interconnected by a single or standard communications protocol.

Some distributed DBMS vendors offer a bridge (gateway) mechanism from their distributed database software to any foreign distributed database software. This bridge (gateway) may be obtained at additional development cost if it has not already been included in the vendor's library of available software.

In the design of a totally new acquisition of distributed DBMS products, it is advisable to consider a mixture of standard conforming DBMSs and communications protocols. Since the technology and products are changing quickly, the designed architecture must be continuously reviewed to prevent being locked into an inflexible mode.

4.3 Considerations for Standards

As the trend towards distributed computing accelerates, the need for standards, guidance, and support will increase. Application distribution and use will be chaotic unless there is an architectural vision and some degree of uniformity in information technology platforms. This is especially true in client/server and workstation environments. To achieve this goal, a systems architecture incorporating standards to meet the users' needs must be established. This architecture must isolate the application software from the lower levels of machine architecture and systems service implementation. The systems architecture serves as the context for user requirements, technology integration, and standards specifications.

The benefits of standardization for both the user and the vendor are many. The number and variety of DDBMS products is increasing. By insisting that purchased products conform to standards, users may be able to choose the best product for each function without being locked into a specific vendor. Thus small to mid-sized vendors may effectively compete in the open marketplace. For effective planning and designing of a DDBMS environment, it is important for the designers to consider what standards already exist and what standards will be emerging in order to be able to incorporate standardized products.

There are many areas of a DDBMS environment in which standards should be applied. Some of the types of standards relevant to the design of a DDBMS include: communication protocols, application programming interfaces, data languages for DBMS, data representation and interchange format, remote data access, etc.

Communication protocol standards are necessary so that systems from different products can connect to a communication network and understand the information being transmitted. An example of a communication protocol standard is the Government Open Systems Interconnection Profile (GOSIP) [FIPS146].

The application programming interface standard is directed towards the goal of having portable applications. This enables software applications developed in one computing environment to run almost unchanged in any other environment. An example of an application programming interface standard is the Portable Operating System Interface for Computer Environments (POSIX) [FIPS151].

The data languages commonly supported by a DBMS are the data definition language, data manipulation language and the data control language. An example of a standard for data language for the relational DBMS model is SQL [FIPS127].

In order to exchange data among open systems, a standard interchange format is necessary. The interchange format consists of a "language" for defining general data structures and the encoding rules. An example of a standard data interchange language is Abstract Syntax Notation One (ASN.1) [ISO87a], [ISO87b].

An important standard for the distributed processing environment is the remote access of data from a client site to a database server site. A specialized remote data access (RDA) protocol based on the SQL standard is currently under development [ISO89].

4.4 Summary of Tasks

To start the overall design process, a review of the organization's existing facilities should be conducted. This review is done to determine whether the new distributed database environment can use some or all of the existing facilities. In deciding to move into a distributed environment, requirements for additional functionalities must be identified. Such organizational issues as setting up regional offices may also be involved. The distributed architecture must take into consideration the actual application to be operating and the characteristics of the user population and the type of workloads to be placed on the system. Such an architecture must also incorporate standardized components.

OVERALL DESIGN OF DISTRIBUTED DATABASE STRATEGY

Summary of Tasks:

o Identify existing, installed facilities that are candidates to be included in the new DDBMS configuration.

o Derive and evaluate several distributed architecture alternatives.

o Establish an overall distributed database environment.

Results:

o Specification for a global distributed database architecture in support of the applications.

Team Members:

o Top Managers, system analysts, DA, DBA and potential users.

Guidelines:

o The design process must incorporate growth trends of the organization to accommodate future nodes.

o Since the technology and products are changing quickly, the designed architecture must be continuously reviewed to prevent being locked into an inflexible mode.

5.0 DETAILED DESIGN FOR DISTRIBUTED DATABASE ENVIRONMENT

The detailed development of a distributed database environment deals with a wide range of issues. This section provides an overview discussion on some of these design issues.

5.1 Hardware Design

The hardware design issue is centered around the client/server model discussed in section 4. The node with the fastest or most powerful computing power is generally assigned the role of server. As server, it will be responsible for most of the processing and data storage. Two important aspects to be considered in choosing server nodes are as follows:

- Processing Power

  The processing power of a server is critical to the response time available to queries; that is, servers should not be allowed to become bottlenecks of the distributed database community. Another concern related to the server's processing power is the fact that there may be other processes competing for processing time other than database requests. Thus, an excessive amount of processing traffic on a server can cause monumental performance problems in the distributed database community.

- Storage Capacity

  The other issue for a server node is its storage capacity. This is critical because the server maintains the central repository for data in the distributed database community. This central repository may be concentrated locally to this one server node or it may be spread across several other remote server nodes.

In comparison to the server node, the client node can be limited in its amount of processing power and storage capacity. Client nodes are typically smaller desk top microcomputers or workstations. The exception to this rule is when a node acts both as a client and a server. Other issues that can dictate the amount of processing power and storage capacity on the client node are the amount of data redundancy kept on the node and the storage requirements for its application software.

5.2 Software Design

For most commercially available distributed DBMSs, the software consists of a family of products that are available on a variety of hardware platforms. A typical family of products might include the following:

- The basic DBMS and its active data dictionary.

- The communication software that is coupled with the DBMS. This type of software may be available with different levels of capability. For example, the minimal capability would be a protocol for remote data access. The next level of capability would be a gateway for remotely accessing foreign databases or files. Foreign databases are those databases established by other brands of DBMS software. The truly distributed functionality would be a communication software product which supports location transparency data accesses and concurrency control. It would also include features such as a two-phase commit protocol for ensuring data consistency.

- Some distributed DBMS vendors also offer additional software utilities such as 4th generation language (4GL), Query-by-forms or Query-by-examples, fancy report writers, and database administration tools for monitoring activities, etc.

One of the first decisions the organization must resolve is the "make-or-buy" decision. To be practical, unless the application is so unique that none of the commercially available distributed DBMSs will suit the needs, it is advisable not to build a home-grown distributed DBMS. Once the organization decides to buy, then a survey and feature analysis of the market needs to be performed. The selection criteria must also take into consideration the amount and the types of software packages that will also be operating within the same platform.

The family of software packages used in a distributed database environment is configured according to the role of each node in the network. For a client node without a database server service, minimum software packages are required such as the communication software and software application tools or languages (i.e., FORTRAN or C). The communication software allows requests to be sent and received by the application software. For a database server node, the minimum software packages not only must have the communication software and the application tools, but also must have a DBMS. For a value-added multi-user operating system environment, a full family of software tools can be configured including fourth-generation languages, two-phase commit protocol, database administrator monitoring tools for tuning, and communication tools for monitoring database requests and traffic within the network.

5.3 Communication Network Design

The linking of computers which are geographically dispersed is accomplished by the communication network. The basic function provided by the communication network is to allow a process running at any site to send a message to a process running on another site of the network.

Factors which need to be considered in selecting or designing the communication network include the following:

- Cost

  The cost of transmitting the message is usually computed by an algorithm that is defined by the system administrator. In general, the cost is proportional to the length of the message and the distance between the source host and the target host. There is always a trade-off between the cost of local data storage and transmitting that data.

- Reliability

  The probability that the message is correctly delivered at its target destination is an important factor to be considered. Reliable transport service with error correction and error detection is currently provided by most communications software.

- Performance

  A critical issue in measuring the performance of a network is the amount of time it takes to deliver a message from its point of origin to its destination. The time required to deliver a message depends on such factors as the amount of traffic on the network, the bandwidth and the capacity of the communications line, how efficiently the communication software can perform optimum routing algorithms, and the mixing of local area network (LAN) with wide area network (WAN).

- Open Systems Interconnection (OSI)

  Standard communication protocols are necessary to allow interoperability among a variety of computer systems without regard to differences in equipment. As of August 15, 1990, it is mandatory that federal agencies must acquire computer network products and services which are in accord with the Government Open Systems Interconnection Profile (GOSIP) [FIPS146].

5.4 Summary of Tasks

The result of the overall design of a distributed database strategy is the determination of the distributed database architecture. Alternatives include the client/server model, the homogeneous DBMS environment, or the truly hetrogeneous distributed DBMS environment. Establishing site requirements involve the identification of the hardware, software and the communication networks for each site.

Hardware and software configurations must be identified for each site that is to participate in the distributed database environment. Decisions on hardware must take into consideration the utilization of existing hardware versus acquiring new hardware.

The selection of distributed DBMS software and communications software must depend on the hardware platform supported. For commercial off-the-shelf software to be utilized, an analysis must incorporate supporting hardware considerations. These hardware and software decisions must be made in a closely integrated manner. For example, it is useless to select one type of hardware, if the desired software cannot function on that particular hardware.

The feature analysis performed for the selection of the products, such as a DBMS, involves identifying the features required for the application, comparing the required features against the features offered by the contending products and making the best final selection decision.

DETAIL DESIGN OF DISTRIBUTED ENVIRONMENT

Summary of Tasks:

- o Identify the hardware platforms for each node participating in the distributed environment.
- o Identify the communications media and the appropriate software for the connectivities of the distributed network.
- o Feature analysis and selection of DBMS and other software support tools to be acquired.

Results:

- o High level specification of the hardware, DBMS, communications, and other support software configuration for each site participating in the DDBMS.

Team Members:

- o Technical system analysts with inputs from top managers and potential uses.

Guidelines:

- o A desirable characteristic is portability of programs, therefore, the distributed environment should permit flexibility to change configurations without creating the need to rewrite programs.
- o Incorporate as many standard products as possible.

6.0 INSTALLATION OF DISTRIBUTED DATABASE ENVIRONMENT

The technical activities performed during the installation phase involve the actual implementation and testing of hardware, software, and communication software for each node of the distributed database environment. Detailed engineering and physical tasks of running cables and establishing hardware configurations will not be described in this report. Summaries of activities for establishing communications and verifying the functioning of software are discussed in the following sections:

6.1 Installation of Hardware

Installing a mainframe computer is very different from installing a microcomputer. If existing hardware is to be employed as a node within the distributed database environment, additional hardware modification may be required. Information on any additional hardware modifications can be determined from the DDBMS technical requirements as specified by its vendor. For example, technical knowledge will be required to add any needed additional random access memory (RAM), communication cards, and communication lines.

The addition of RAM may take the form of installing memory chips into a memory board or purchasing a memory board with the memory chips already installed. The appropriate existing memory cards must be identified and extracted from their position in the computer's chassis or cabinet. Any new card or existing cards with the required added-memory must then be installed in the computer chassis. Depending on the targeted computer, switches or jumpers on the memory cards may have to be set for proper recognition of the added memory.

After adding any required memory, diagnostic software should be run to confirm that the computing software recognizes this additional memory. In most computers, there is built-in firmware that will check for the existence of the additional memory during the initial boot process of the computer.

To establish connection among computers targeted as nodes in the distributed database environment, a communication card must be present in each computer. The required features for a communication card should be confirmed by the vendor of the candidate DDBMS software. It is advisable to purchase the compatible communication card used by the vendor in developing the DDBMS software. The brand of the communication card and the computing platform will determine any extras that will be required to connect to a network. For example, on personal computers, the communication cards will generally allow either an external transceiver connection (a thick-net cabling connection) or connection to its built-in transceiver (a thin-net cabling connection). In the first case, an external transceiver will have to be purchased in addition to the communication card. In the second case, only thin-net cabling will be required. On workstations and other larger computing platforms, the first case is generally the only option.

After installing the communication card, the following tasks need to be performed:

- o Run the hardware diagnostic software, and
- o test the remote connectivity capability.

Running the hardware diagnostic software will verify system recognition of the newly installed communication card. Proper recognition by the system means that the communication card is linked to an acceptable address (non-conflicting) and its features have been recorded by the system.

It is highly desirable to confirm remote connectivity before attempting to install the DDBMS software. This will eliminate having to consider local hardware communication problems should connectivity problems arise after installing the DDBMS software.

6.2 Installation of the Network Communication Software

The primary task of the installation of the network communication software involves establishment of specific communication parameters. These parameters are generally specified during the initial setup process and are contained in various special parameter files. The names and locations of these parameter files are known to the communication software. For example, one such parameter file is generally called a HOST file. This file usually contains specific internet addresses and associated logical names for locating nodes that are on the local area network (LAN) and the related wide area network (WAN).

In addition to the HOST file, there is generally a SERVICES file which contains the address of the communication port that is to be used by the DDBMS's communication software. Thus, when the installed network communication software reads the SERVICES file, the designated communication port will be reserved as a communication link to the DDBMS's communication software.

It is critical that the DDBMS communication software and the installed network communication software for the operating system have the same protocol. For example, if the network communication software uses TCP/IP protocol, the DDBMS communication software needs to know how to package communication requests accordingly. If the network communication software uses GOSIP, the DDBMS communication software requests need to be packaged according to the appropriate layers of the ISO/OSI seven layer model. Discussions of these layers and their protocols are beyond the scope of this document. More information on GOSIP can be found in [BOLA89] and [BOLA90].

6.3 Installation of the DBMS Communication Software

No distributed database management system can function in a network environment unless it is accompanied by the appropriate DBMS communications software. When purchasing the DBMS software, it should be confirmed with the vendor whether the communication features are bundled with the basic DBMS package or are separate features.

There must be compatibility between the network communication software and the DBMS communication software that interfaces with it. After assuring compatibility, the next step in the installation is to identify to the DBMS communication software its required communication parameter file. It is through such a parameter file that the DBMS communication software will have knowledge of remote DBMS systems.

6.4 Installation of the DDBMS and Other Support Software

The DBMS kernel has to be installed before installing the DDBMS. Installation of the DBMS kernel and its support software, such as 4th generation languages, report writers, forms processors, graphics packages, etc. follow the same general procedures:

- o Preparation - This task requires super user authority to first allocate memory space. Generally a separate directory is established in preparation for loading the software.
- o Loading - The initial parameter file is loaded into the memory.
- o Configuration - The configuration of the shared memory is adjusted in accordance with the requirements of the chosen architecture.
- o Compilation - The loaded software, in source form, is compiled to create an absolute module ready for execution.
- o Testing and Verification - The software is executed to test and verify it for correctness.
- o Cleanup - Delete any unwanted files.

After the DBMS kernel and its support software has been individually installed, there is the task of identifying to the DBMS kernel a set of logical definitions that will delineate local and remote data. It is through this association that the support software is capable of transparently accessing data throughout the global DDBMS schema.

Based on the design considerations, each node may have a different set of software support requirements. Therefore, the installation will include only those support software packages as required.

6.5 Summary of Tasks

After the decision has been made on the equipment necessary for each node, the acquisition process should begin. The plan for installation should be coordinated such that each node will be installed and checked out, followed by the installation of the networking between nodes.

The installation phase requires very skilled technical engineers and system programmers.

INSTALLATION OF THE DISTRIBUTED ENVIRONMENT

Summary of Tasks:

o For each node, install hardware and software.

o Install local area network (LAN) or wide area network (WAN) and then install the appropriate communication software.

o Integrate and test the total environment.

Results:

o The distributed environment properly installed and functioning.

Team Members:

o Skilled hardware, software and communications engineers.

Guidelines:

o To ensure proper installation, each node and each subsystem must be checked before total system checking for proper functioning.

7.0 DETAILED DESIGN FOR DISTRIBUTED DATABASE APPLICATION

Once the distributed environment is in place, the design for an application that will run under the environment can be finalized. Although the tasks of establishing the environment and the application may seem to be separate, the planning and design phases for each of them must incorporate all requirements.

Many distributed database application design issues were addressed in [FONG85], [TEOR82]. In many cases, distributed database design issues are still active areas of research. Some of these issues are discussed below.

7.1 Application Database Architectural Alternatives

The needs of the organization's users must be evaluated in order to decide which of the different database architectural alternatives is best for the organization. A database architecture is the way in which data is integrated with respect to an application. The alternatives include databases that are logically centralized but physically dispersed, or databases that are loosely-coupled (or "federated") with each site having local autonomy.

7.2 How to Distribute the Databases

Two of the most critical decisions in distributed database design are how to partition the data into different fragments and how to introduce replication of data in order to improve the performance and reliability of the system.

There are documented techniques for distributed database design that exist in the research literature today [TEOR89], but, in practice, the allocation of data is often very ad hoc. Some of the current approaches to distributed database design consist of: top-down, bottom-up, backward-forward, and activity analysis.

The top-down approach views the organization's data as a whole and decomposes the data into various nodes which are physically dispersed. The decision concerning where to put data is determined by data entry collection stations. For example, an organization may have several regional offices, and each regional office collects and maintains its own regional data. However, only one unified global schema exists.

The bottom-up approach views each node having databases as a processing entity. This occurs when existing databases are integrated into a single global schema, but each database is semantically treated as a whole. This is the same as the federated approach.

The backward-forward (output driven) approach begins with the identification of the output of the application and subsequently determines where the data sources are to be placed.

The activity analysis (process driven) approach begins with the identification of both manual and automated processes. For each process, the input and its source are determined. Additionally, the output produced and their destinations are identified. This information is then used to determine the distributed architecture.

Data Fragmentation

Almost any distributed database design will require decisions concerning how data is to be fragmented on several different nodes, or where data is to be replicated on several different nodes. The fragmentation of databases usually applies to relational databases, in which the data are viewed as tables consisting of rows and columns. The fragmentation can be vertical or horizontal.

Vertical fragmentation involves splitting a table, column-wise, into several smaller tables with fewer columns, which are then distributed to different nodes. Horizontal fragmentation involves splitting a table, row-wise, into several smaller tables distributed at several different nodes. To fragment a table horizontally involves storing different rows of data at different sites depending upon the needs of the application.

The decision to fragment the data is usually done heuristically by analyzing each relation in term of the data volume being retrieved and noting where the requests originate. A "best fit" method can be found in [CERI84].

Data Replication

Redundantly stored data is referred to as replicated data. The purpose of replicating data is to permit frequent and faster use of the same data by multiple sites. However, allocating additional copies of data not only uses more storage space, it also increases the complexity and the time needed to perform updates. The decision whether or not to replicate data must include a consideration of the costs and benefits for each site versus the entire organization. The benefit at a specific site is measured by taking the difference in cost between doing a remote query versus a local query, if the data is replicated locally. This figure must then be compared against the cost of maintaining multiple accurate copies of the replicated data.

The other consideration for replicated data is the "availability" constraint. Some data may be deemed of such high value that it must be available at all times. Accordingly, a crash at one site should not cause the loss of data to other sites that may need the data now.

7.3 The Role of the Data Dictionary/Directory for DDBMS

The data dictionary/directory (DD/D) for the distributed database system plays the crucial role of logically integrating all the physically dispersed databases. There are two levels of DD/D schema for the distributed environment. The first is the DD/D which contains all the schema information for the whole network. This is known as the global schema. The second is the local DD/D that contains the schema information pertaining to the databases located at that individual node. This is known as the local schema.

Global Schema

The global schema, as maintained in the DD/D is used to support a number of different applications. These applications in a distributed DBMS environment include all of the services normally satisfied by the global schema in a DD/D for a centralized environment, plus the following additional functional requirements:

- o identify data location in the network.
- o support coordination of distributed data retrieval, consistency, and integrity especially in the case of replicated data.
- o support data translation for user processes if databases within the network are of different data models.
- o support source and target metadata descriptions to be used for the export and import of metadata and data.

A sophisticated global schema also must have the capability to resolve data incompatibilities. Examples of data incompatibilities include: scale or unit differences, different naming or encoding methods for semantically equivalent data.

Local Schema

Some of the functional requirements that must be satisfied by the local schema that is maintained in a local data dictionary are:

- o provide accessing of data.
- o support security rules.
- o provide the presentation of data on screen or in reports.

A sophisticated local schema may also have statistical information that can be used in optimizing access and performance.

Where to Place the DD/D

To a certain extent, the same considerations that apply to the distribution of application systems data also apply to the distribution of the DD/D and its schema specifications of the distributed databases. The options for coordinating metadata across nodes of the network include:

- o A centralized, or global schema, located at one single node. Nodes in the network must poll this global schema to find the desired data.
- o Distributed replicated schemas in which a copy of the global schema is located at each node, containing information concerning the data in the entire network.
- o Distributed partitioned schemas, in which each node has its own local schema. If a request comes in for data not in that node, the user or the system must query all the other nodes until the desired schema is found.
- o Hierarchy of schemas in which a "master" copy of the global schema may be located at some nodes, and each local node maintains a local schema.

Each of these options has advantages and disadvantages in terms of availability, performance, and cost. Each organization establishing a distributed database application must select the best approach with respect to the nature of the application. In today's commercial products, the definitions for both global and local schemas coexist on the same computing equipment. Such coexistence permits site autonomy.

7.4 Schema and Data Update Control

When to update, where to update, and how to update data and metadata are complex design issues. Completing an update transaction in a distributed database environment is technically a complex issue, since at the same time data may be getting updated, one or more users might also be performing a query against the same data. This problem is even more complex when data are replicated since updating multiple copies of the same data while keeping the data consistent requires special concurrency control mechanisms.

Distributed concurrency control mechanisms, at present, are still very much a research issue. One way of ensuring data consistency, which is currently supported in some DDBMS, is an algorithm called "two phase commit." Two phase commit is a protocol which first sends a message to all involved sites to "prepare" to commit an update. When all sites involved acknowledge their readiness, then the second message is sent to "commit" the update.

7.5 Data Administration Functions

For a distributed database application to be successful, it must be recognized right from the beginning that the overall management of the data resources within the distributed environment is a critical task. The guide to the data administration function for a centralized database environment appears in [ROSE89]. The data administration function for a distributed environment needs to be built based on the fundamentals specified in [ROSE89] and then extended to cover the additional requirements generated by having multiple locations where data are stored.

The data administration function can be perceived of as many different roles with various job responsibilities. Depending upon the size of the organization, these functions may be performed by one or more persons. There are different terms, or job titles, associated with these roles. Some of these roles are described as follows:

o Top-level data strategist/administrator

  For a distributed environment, this position manages and controls the total corporate-wide data. The responsibilities include deciding what data resources should be installed within the distributed environment, determining who will use the database, establishing data standards, and controlling global data dictionary access and accuracy.

o Distributed database administrators (DBA)

  For each node within the distributed environment which acts as a database server, there might be a database administrator role. The responsibilities assigned to this DBA concern the management and control of the database segment resident at this node. The role of distributed DBAs may include functions such as data analysis and modeling, along with enforcement of policies and standards established by the top-level data strategist/administrator. Other technical functions include maintaining integrity and consistency of the resident segment of the distributed database, and resolving conflicts of incompatible data, both at the local site and in relation to data available at other sites.

o Network administrator

  The responsibilities of the network administrator are to manage the distributed networks, ensure smooth communications between nodes, and provide maintenance and support to the network users.

For the creation and control of the data resources in an organization, the methodology could be basically said to fall somewhere between the following two extremes:

- o Totally centralized control by top-level data administrator. This individual sets the database policies and standards for the accessing and maintenance of all corporate-wide databases and data dictionaries installed within the distributed environment.

- o Totally decentralized with no overall policies and standards for the accessing and maintenance of data. Each subgroup or each node's database administrator owns the database at the node and each is free to establish their own local database administration policies.

7.6 Summary of Tasks

The design of an application should begin early on when the organization has decided to migrate to distributed database processing. The first application may just be a demonstration prototype to test and verify the correctness of the distributed database environment.

The design, implementation and support for a large scale distributed application requires extensive procedures and controls, which is beyond the scope of this guide. This guide discusses tasks for the development of a small demonstration application for the testing of the distributed database environment.

After completing the requirements analysis for the application, a logical database design need to be conducted. A step-by-step procedure for doing logical database design for a centralized database is given in [FONG85]. The logical database design activities consist of the following two main tasks:

- o Data flow analysis by identifying each entity and its relationships with other entities.

- o Process flow analysis by identifying each function and procedure of the operations.

Next, the possible alternatives as to where to distribute the data need to be analyzed, based upon the available locations of the distributed nodes and the accessing requirements of the applications. Additionally, during the design of distributed database applications, one must also consider where to place the global schema and how to resolve the data update control problems.

For a large scale distributed database application, an important administrative task is the establishment of appropriate data administration (DA) functions. This decision has tremendous implications as to who owns and maintains the databases at each node.

DETAILED DESIGN FOR DDB APPLICATIONS

Summary of Tasks:

- o Perform requirement analysis.
- o Establish Logical database design.
- o Resolve database and metadata distribution issues.
- o Establish DA and DBA functions.

Results:

- o Specification of conceptual schema and data design.
- o Specification of application system.

Team Members:

- o Technical system analysts, potential users, DA and DBAs.

Guidelines:

- o Detailed planning for data distribution must consider the data usage pattern.
- o End users must be consulted in the design of applications.

8.0 APPLICATION DEVELOPMENT

The actual coding and debugging of the application system occurs during this phase and is based on the results of the detailed design of the application. For application development on the distributed database, the processes are summarized as follows:

8.1 Define Schemas

The definition of schemas at each node is the process of creating an empty container based on a given structure. Different DDBMS dictate various methods for data definitions. If the DDBMS is a relational DBMS, then defining data definitions is referred to as creating tables. The issues to be considered are:

- o Identification of needed fields for each table at each node, and their associated data types.
- o Ensuring the naming conventions for each field. A guide to naming conventions appears in [NEWT87].
- o Key fields for the purpose of linking to other tables need to be identified.
- o Schema constraints and data type checking functions need to be implemented as triggers.
- o Define access control restrictions and establish global and local authorities required to define or modify access control restrictions.

8.2 Populate Database

After the data definitions are established for each node, the data is ready to be populated. There are various ways for the DDBMS to accept data. Large quantities of data are typically bulk loaded into the database server. If there are existing databases or files to be loaded into the newly established DDBMS, utilities such as export/import or text file loaders can be used. Some DDBMS provide application development tools, such as forms processing. This method permits forms or screens to be created so that data may be entered record-by-record.

8.3 Application Program Construction

Certain functions require construction of the application programs using a host programming language, e.g., FORTRAN or C, etc. Coding, debugging, and testing are performed by programmers.

8.4 Multi-site Requests Processing

To illustrate multi-site query or update request processing, the scenario depends on whether the requestor is an application programmer or an end-user. For the end-user illustration, a request is issued at node A. If the DDBMS supports location transparency, then this request is analyzed at node A to determine where and how to fetch the data. The request is then executed based upon the most optimal way of retrieving the required data. The answers are then assembled and displayed to the requestor at node A. For an application programmer, the global dictionary needs to be consulted and links need to be established for the processing of the multi-site request.

Solving the problem of choosing the most efficient access method and guaranteeing data consistency are the responsibilities of the application programmers and database administrators.

8.5 Testing and Verifying Correctness

Testing and verifying correctness involves writing test cases, executing each test case, and verifying that the results obtained are expected answers. For the distributed database application, testing need to be performed at two levels. For each node, test cases must be performed to ensure that the application is functioning properly. Then, at the next level, test cases must be performed involving multi-sites to ensure that the distributed application, as a whole, is functioning correctly.

8.6 Summary of Tasks

For a large scale application project, the tasks for the installation and implementation of the distributed database application can be very complex requiring knowledge of software engineering disciplines.

The application implementation phase is typically performed by programmers. The tasks involved are actual hands-on coding or generation of database schemas and manipulation of the database using the DBMS vendor supplied utilities and tools. For a distributed application, the final debugging and testing involves not only each single site functioning correctly, but multi-site requests and data manipulations must be tested to ensure that all the components of the entire distributed database system are working together in an integrated manner.

APPLICATION DEVELOPMENT

Summary of Tasks:

- o Define schemas for all database servers.
- o Populate databases with application data.
- o Test requests that operate on single site and multi-sites.
- o Write and test application code if necessary.

Results:

- o Operational distributed application databases and programs.

Team Members:

- o Skilled database designers and programmers.

Guidelines:

- o The data definition and application programs should be well documented to avoid confusions.
- o Build as many data validation procedures as needed to ensure data quality and consistency

9.0 SUPPORT FOR THE DISTRIBUTED DATABASE

The final stage occurs when the distributed environment and the distributed application system are ready to be released for real operational use. It is at that time that the support and maintenance tasks begin. In practice, it is usually the case that the support for the distributed database environment and the support for the application will be done by the same team of system programmers and database administrators. However, some organizations may prefer to separate the tasks of supporting the environment and supporting the applications.

Overall system management and support for the distributed database environment involves a team of database administrators. If the distributed environment includes several locations which are physically apart, then it is desirable to have a local database administrator and a network data manager at each location. The network data manager is sometimes referred to as the global database administrator. The local database administrator manages the local node support activities. The global database administrator coordinates the communications activities and provides services to other distributed nodes.

9.1 Tuning for Better Performance

One of the main tasks involved in supporting the distributed database system is to monitor the traffic within the network and tune the network systems for better performance. There are various tools supplied by DBMS vendors for gathering system statistics. Tuning a distributed environment is much more complex than tuning a centralized system because, not only must monitoring be conducted at each local node, but it must also be performed at the network interface level across the entire distributed environment.

9.2 Backup, Recovery and Security Protection

When the distributed DBMS system and the network software encounter system failure, such as hardware or network problems, the prompt restoration to proper functioning is a crucial task to ensure that reliable service is provided to the users.

Providing both physical security and software protection against unauthorized users are two of the support activities to be performed by the local and global database administrators. Since security is an extensive and complex subject, it is beyond the scope of this report except to say that security requirements need to be considered at every step in the development of distributed database systems. See computer security publications issued by NIST [NIST90].

9.3 User Training and Resolving Problems

Another important task for the support phase is to provide users with adequate training on the use of the distributed database system. This task could also include the assignment of user account and password numbers, allocating working and permanent storage, etc.

When DBMS vendors put out new versions of their software, it is the responsibility of the local database administrator to install the new version. When problems are detected, the support staff must verify and isolate the problem. The problem should either be fixed or should be reported to the appropriate vendors.

9.4 Summary of Tasks

The support task is of a continuous nature. Therefore, one or more permanent system programmers or engineers should be on duty at all times. The skills of the support team members and the size of that team will depend to a large degree on the size and nature of the distributed database environment and the applications.

SUPPORT OF ENVIRONMENT AND APPLICATIONS

Summary of Tasks:

- o Tuning of better performance.
- o Backup, recovery and security protection.
- o User training and resolving problems.

Results:

- o Distributed database environment properly maintained and operational.

Team Members:

- o System managers, system programmers, local global database administrators.

Guidelines:

- o Support staff must be very technical as well as possess interpersonal skills.
- o Designate responsibilities for support at geographically separated locations.

10.0 CONCLUDING REMARKS

This guide describes a complete life cycle development for migrating to a distributed database environment, and the development of a prototype distributed application. Organizations may use any of the strategies described, or a combination of these strategies.

The strategies presented are generic. Their advantages and disadvantages will depend on the combination of an organization's internal policy and the proposed applications. Some factors that must be considered include: cost, simplicity of implementation, size and scope of the distributed application, and compatibility with current hardware and software design.

10.1 Critical Success Factors

As it is pointed out in earlier sections of this guide, the development of a distributed database environment and distributed applications will only be successful when each phase of the development is carefully planned and executed. The success of the development of a distributed environment and applications is difficult to quantify. Different measures are appropriate for different application areas. Some commonly perceived measures of success are listed as follows:

- o Adequate performance of the overall distributed DBMS applications.

- o Users are more satisfied with the newly installed distributed application than the previous non-distributed application.

- o Lower costs for resources and better efficiency in the processing of data with the distributed applications.

- o Improved maintenance of the distributed database system and the distributed data administration procedures.

10.2 Future Issues

The development of such a project from conception to operation requires a long time span. Some large scale distributed environments and applications could take more than 2 years to complete. However, during the 2 year development period, distributed database technology and the organization's requirements would be changing. This situation does, of course, present the possibility of having an obsolete system upon completion. A few observations follow:

- The technology of the truly distributed, all-powerful, all-embracing data model and language to provide transparent accesses to separate databases and computers is still not as mature as one might expect. All the present distributed environments in support of operational distributed database applications are either homogeneous, or federated with translation mechanisms.

- Issues of access control, concurrency control, schema and transaction management, performance, capacity growth, support for complex environments, etc. are still in the research stage. In practice, distributed data processing is still partly a learning process.

- In current practice, there is increased user dependency on single vendors to supply the DBMS and communications products. Some very difficult decisions will have to be made by the system designers in order to achieve the best balance of functionality, performance, and vendor independence. The use of standard products will alleviate the vendor dependence problem.

- In the near term, the release of a RDA/SQL standard will provide technical specifications on how to package SQL network service requests. These packages will contain service elements for:

    - association control, which includes establishing an association between the client and server remote sites and managing connections to specific databases at the server site,

    - transfer of database operations and parameters from client to server, and the transfer of resulting data from server to client,

    - transaction management which includes capabilities for both one-phase and two-phase commit.

  The introduction of the RDA/SQL standard will pave the way for wider interoperability among heterogeneous DDBMS nodes supporting SQL data languages. A consortium of SQL vendors known as SQL Access Group hopes to demonstrate interconnection of working prototypes in 1991.

- The migration from a centralized environment to a distributed environment will require careful planning. A more important consideration is that data sharing and resource consolidation will require cultural changes.

Some of the open questions as listed above cannot be answered until more experience is gained and documented. The "lessons-learned" in such a project will be very valuable. The authors for this guide would welcome any further guidelines and comments on the development of distributed database systems.

11.0 REFERENCES

[BOLA89] Boland, T., Government Open Systems Interconnection Profile Users' Guide, NIST Special Publication 500-163, August 1989.

[BOLA90] Boland, T., Stable Implementation Agreements for Open Systems Interconnection Protocols, Version 3 Edition 1, NIST Special Publication 500-177, March 1990.

[CERI84] Ceri, S. and Pelagatti, G., Distributed Databases: Principles and Systems, McGraw Hill, 1984.

[FINK89] Finkelstein, Richard, "Client/Server Computing - The Best of Two Worlds," CONNECT Magazine, Summer 1989. pp. 24-27.

[FIPS127] Federal Information Processing Standards Publication 127-1, Database Language SQL, National Institute of Standards and Technology, February 1990.

[FIPS146] Federal Information Processing Standards Publication 146, Government Open Systems Interconnection Profile (GOSIP), National Institute of Standards and Technology, 1989.

[FIPS151] Federal Information Processing Standards Publication 151-1, POSIX: Portable Operating System Interface for Computer Environments, National Institute of Standards and Technology, 1989.

[FONG85] Fong, E., et al., Guide on Logical Database Design, NIST Special Publication 500-122, February 1985.

[FONG88] Fong, E. and Rosen, B. K., Guide to Distributed Database Management, NIST Special Publication 500-154, April 1988.

[HEIL89] Heiler, S., "The Integration of Heterogeneous Computing Environments," in chapter 6 of Fong, E. and Goldfine, A. (editors), Information Management Directions: the Integration Challenge, NIST Special Publication 500-167, September 1989.

[HEIM85] Heimbigner, D. and McLeod, D., "A Federated Architecture for Information Management," ACM Transactions on Office Information Systems, Vol. 3, No. 3, July 1985, pp. 253-278.

[ISO87a] ISO 8824 - Information Processing Systems - Open Systems Interconnection - Specification of Abstract Syntax Notation One (ASN.1), Reference Number: ISO 8824:-1987(E).

[ISO87b] ISO 8825 - Information Processing Systems - Open Systems Interconnection - Specification of Basic Encoding Rules for Abstract Syntax Notation One (ASN.1), Reference Number: ISO 8825:1987(E).

[ISO89] ISO "Information Processing Systems - Open Systems Interconnection - Remote Database Access - SQL Specialization," ISO/IEC JTC1/SC21 N4281 Committee Draft 9579 - Part 2, 1989.

[MART81] Martin, J., <u>Design and Strategy for Distributed Data Processing</u>, Prentice-Hall, Inc. Englewood Cliffs, New Jersey 07632, 1981.

[NEWT87] Newton, J. J., <u>Guide on Data Entity Naming Conventions</u>, NIST Special Publication 500-149, October 1987.

[NIST90] NIST Publications List 91, <u>Computer Security Publications</u>, NIST, Revised March 1990.

[ROSE89] Rosen, B. K. and Law, M., <u>Guide to Data Administration</u>, NIST Special Publication 500-173, October 1989.

[TEOR82] Teorey, T. J., and Fry, J. P., <u>Design of Database Structure</u>, Prentice-Hall, Inc., Englewood Cliffs, N.J. 07632, 1982.

[TEOR89] Teorey, T. J. "Distributed Database Design: A Practical Approach and Examples," in <u>SIGMOD Record</u>, Vol. 18, No. 4, December 1989.

APPENDIX A - DESCRIPTION OF DISTRIBUTED ENVIRONMENT

The distributed database environment of the NIST/CSL Database Laboratory used two commercial DDBMS: ORACLE and INGRES.

The **ORACLE** distributed node configuration was as follows:

NODE A: A server node on a minicomputer.

<u>Hardware</u>	<u>Software</u>
VAX 11/785	Oracle version 5.1.22
Ethernet Communication Card	VMS version 4.0 or later
Thin net transceiver and cable	Wollongong TCP/IP
2MB memory (recommended)	SQL*NET TCP/IP
33,000 blocks of disk space	

NODE B: A server node on a microcomputer.

<u>Hardware</u>	<u>Software</u>
Compaq 286	Oracle version 5.1.17
Excelan Communication Card (205T)	Xenix System V, v2.1.3
Thin net cable	(Santa Cruz Operation)
3MB memory (recommend)	Excelan 8011-03 TCP/IP
3MB of swap space	SQL*Net TCP/IP
25,000 blocks of disk space	

NODE C: A client node on a microcomputer.

<u>Hardware</u>	<u>Software</u>
Compaq 286	DOS 3.0 or higher
Excelan Communication Card (205T)	Excelan 8011-03 TCP/IP
Thin net cable	SQL*Net TCP/IP
640K memory	
4.5MB of disk space	

NODE D: A client node on a MacIntosh.

<u>Hardware</u>	<u>Software</u>
MacIntosh II	Multi-Finder
Kinetics Communication Card)	Kinetics TCP/IP
Thin net cable	SQL-Net TCP/IP
2MB memory	HyperCard Version 1.2
5MB of disk space	MacIntosh Programmers Workshop - optional

For the **INGRES** distributed environment, the configuration was as follows:

NODE A: A server node on a minicomputer.

Hardware	Software
VAX 11/785	Ingres 6.3
Ethernet Communication Card	VMS version 5.0 or later
Thin net transceiver and cable	Wollongong TCP/IP
8 MB memory (recommended)	Ingres*Net TCP/IP
55 MB of disk space	

NODE B: A server node on a workstation.

Hardware	Software
SUN 386i	Ingres 6.2
SUN Communication Card	SUN Unix version 4.0.2
Thin net transceiver and cable	SUN TCP/IP
8 MB memory (recommended)	Ingres*Net TCP/IP
55 MB of disk space	

APPENDIX B - DESCRIPTION OF THE DISTRIBUTED APPLICATION

PROTOTYPE PROJECT OVERVIEW

The distributed database application uses an existing database file established for the tracking of the Division's Property in terms of hardware and software equipment. This file, which is continually maintained, has over 20 data elements, with over 300 records for the hardware equipment and over 400 records for the software.

The application was defined for both the ORACLE distributed environment and the INGRES distributed environment. A number of steps were required to produce a distributed database application. Some of these steps were similar with both Oracle and Ingres and some of the steps were different.

DESIGN OF THE APPLICATION

The first steps were similar for both ORACLE and INGRES. These included identification of the problem, design considerations for the necessary tables, deciding where the tables would be located, and what fields were needed.

For the division property application, five data elements were chosen to serve as keys throughout all of the distributed database tables. The horizontal fragmented technique was chosen for dividing the database across the different nodes. Five tables were identified and installed on five different nodes. These nodes were created by using five different usernames.

DATA DEFINITIONS AND POPULATING THE DATABASE IN ORACLE

To make each table unique, different data values or records were entered into different tables. To accomplish this, a distributed database table was created using SQL. Fields and records that were needed from the big existing database file were spooled off into a "dummy file." This file was then loaded into the newly created distributed database table by using the SQL-loader tool. After the newly created distributed database file had the records loaded in, the other tables were created on the other nodes and the data was copied in by using the following copy command:

```
COPY FROM NLHARD/NLHARD@T:icst-ise:P to GLHARD/GLHARD@T:penguin:P
    CREATE GL_HARD -
    USING SELECT * FROM NL_HARD WHERE NAME = 'GRAPHIC LAB';
```

In the above example, on the first line, the table is being copied from the VAX (**icst-ise**) with the username of **NLHARD/NLHARD** to the XENIX (**penguin**) with the username of **GLHARD/GLHARD**. On the second line, a new table is being created, entitled : **GL_HARD**. On the third line, a select statement is being used to bring over to the new table only those records that deal with the graphics lab. This command allows the user to create the table and load the necessary records on a different node without having to physically go to that node to do the work.

After all the tables were created and had their records loaded in, the tables were reviewed and modified so as to finalize what records were located at each individual table.

DATA DEFINITIONS AND POPULATING DATABASE IN INGRES

For the Ingres application, the tables were created using the COPY command shown below. The records were then loaded from a "text file" into the appropriate tables.

```
COPY TABLE ALL_HARDWARE
(NBS=C7,I=C2,ITEM_NAME=C26,OWNER=C11,ROOM=C0NL)
FROM 'ALL_HARDWARE.LIS'
```

In the above example, the table **ALL_HARDWARE** had already been created by the usual SQL command. Also, the data being used for this table is the same data used in the ORACLE database. The data was spooled from the ORACLE database into a file called **ALL_HARDWA-RE.LIS**. The file was then loaded into the INGRES table by using the above command. This command copied into table ALL_HARDWARE on a row-by-row basis data value occurrences for the respective attributes "NBS", "I", "ITEM_NAME", "OWNER", AND "ROOM" from the file A::_HARDWARE.LIS.

DEFINED TABLES IN ORACLE

TABLE NAME	RECORDS USED	USERNAME	NODE	OS
NL_HARD	No lab records	NLHARD	VAX	VMS
VALL_HARD	Val lab records	VALLHARD	VAX	VMS
KBL_HARD	KB lab records	KBLHARD	COMPAQ 286	DOS
DBL_HARD	DB lab records	DBLHARD	COMPAQ 286	XENIX
GL_HARD	Graph lab records	GLHARD	COMPAQ 286	XENIX

DEFINED TABLES IN INGRES

TABLE NAME	RECORDS USED	USERNAME	NODE	OS
ALL_HARDWARE	All hardware	DIV_HARD	VAX	VMS
EXCEPT_LAB	No lab records	DIV_HARD	VAX	VMS
GRAPH_ONLY	Graph lab records	DIV_HARD	SUN 386i	XENIX
TAB_SOFT	All software	DIV_HARD	SUN 386i	XENIX

The data element names used as keys across the distributed nodes are:

```
NBS NUMBER
SERIAL NUMBER
ITEM NAME
OWNER
LOCATION
```

MULTI-NODE REQUEST PROCESSING

When dealing with multiple tables and multiple nodes, database links, synonyms, views, and a command called "JOINDEFS" were very helpful. They provide the capability to query a table on a node that is remote, or different, from the current node on which the user is operating.

ORACLE DISTRIBUTED QUERY EXAMPLE:

In Oracle, **database links** were used to represent the username, the password, and the network name. To access a desired table at a remote node, the name of that table has to be linked to a designated database link name. Below is a sample of the database link command:

```
CREATE DATABASE LINK {database link name}
CONNECT TO {username}
IDENTIFIED BY {password}
USING '{network name}';
```

This is an example using the above command:

```
CREATE DATABASE LINK GLLAB
CONNECT TO GLHARD
IDENTIFIED BY GLHARD
USING '@T:penguin:P';
```

A query using a database link might look like this:

```
SELECT * FROM DBL_HARD@DBLAB
```

In INGRES a distributed database definition was used. Within this distributed database definition, the INGRES tables and nodes were registered. Below is an example of how a table and node is registered with a distributed database name:

```
CREATEDB {distributed database name}
REGISTER TABLE {table name}
AS LINK WITH NODE = {node name},
DATABASE = {database name}
```

This is an example using the above command:

```
CREATEDB DDPROPERTY/D
REGISTER TABLE ALL_HARDWARE
AS LINK WITH NODE = 'VAXNODE',
DATABASE = 'DIV610'
```

(NOTE: Tables that are not registered are not known to the distributed database; therefore, they are not accessible by remote users.)

In Oracle, a **synonym** was used as a tool to represent the combination of the table name and the database link name. This tool was used because of its simplicity. Below is an example of how a synonym is created:

```
CREATE SYNONYM {synonym name}
FOR {table name} @ {database link name};
```

This is an example using the above command:

```
CREATE SYNONYM GL_LAB
FOR GL_HARD@GLLAB;
```

A query using a synonym might look like this:

```
SELECT * FROM DBL_LAB
```

To bring all the information from all the tables together in Oracle, a **view** was used. Shown below is an example of how a view is created:

```
CREATE VIEW {view name} AS
SELECT {column name 1,column name 2,etc.} FROM {synonym}
UNION
SELECT {column name 1,column name 2,etc.} FROM {synonym}
```

This is an example using the above command:

```
CREATE VIEW ALL_HARDWARE AS
SELECT NBS,SERIAL,ITEM_NAME,NAME,ROOM FROM NO_LAB
UNION
SELECT NBS,SERIAL,ITEM_NAME,NAME,ROOM FROM VAL_LAB
```

INGRES DISTRIBUTED QUERY EXAMPLE

In INGRES, the tool called the **JOINDEF**, which is similar to the **VIEW** was used. Queries, updates, and deletes can be done in the JOINDEF. However, only queries can be done in the view.

A JOINDEF is done within INGMENU and a distributed database name (in this project DDPROPERTY/D) is used. After selecting the JOINDEF procedure, menus appear which ask a series of questions. For example, the series of questions could include: the name of the JOINDEF, its relationship role (i.e., ONE-TO-ONE relationship or a ONE-TO-MANY relationship), and the list of selectable columns. The tables are then connected by the 'key field'.

ORACLE GLOBAL DATA DICTIONARY

The multi-node request processing is made possible by the established global data dictionary. The global data dictionary resides at all database server nodes. The table names of the ORACLE global data dictionaries are as follows:

TABLE_NAME	DB_LINK	SYNONYM	USERNAME	NODE
NL_HARD	NOLAB	NO_LAB	NLHARD	VAX/VMS
VALL_HARD	VALLAB	VAL_LAB	VALLHARD	VAX/VMS
KBL_HARD	xxxxx	xxxxxxx	KBLHARD	COMPAQ/DOS
DBL_HARD	DBLAB	DBL_LAB	DBLHARD	COMPAQ/XENIX
GL_HARD	GLLAB	GL_LAB	GLHARD	COMPAQ/XENIX

(NOTE: Because the compaq 286 using DOS is set-up as a client, and not as a server, a database link or synonym could not be created for the username KBLHARD.)

INGRES GLOBAL DATA DICTIONARY

The multi-node request processing in INGRES is made possible by the global data dictionary residing at the database server node. The table names of the INGRES global data dictionaries are as follows:

TABLE_NAME	DB_DIFF	JOINDEF	USERNAME	NODE
ALL_HARDWARE	DD/PROPERTY	ALL_HARD	DIV_HARD	VAX/VMS
EXCEPT_LAB	DD/PROPERTY	NO_LAB	DIV_HARD	VAX/VMS
GRAPH_ONLY	DD/PROPERTY	GRPH_LAB	DIV_HARD	SUN/XENIX
TAB_SOFT	DD/PROPERTY	ALL_SOFT	DIV_HARD	SUN/XENIX

NIST-114A (REV. 3-90) **U.S. DEPARTMENT OF COMMERCE** NATIONAL INSTITUTE OF STANDARDS AND TECHNOLOGY **BIBLIOGRAPHIC DATA SHEET**	1. PUBLICATION OR REPORT NUMBER NIST/ Sp 500-185
	2. PERFORMING ORGANIZATION REPORT NUMBER
	3. PUBLICATION DATE February 1991

4. TITLE AND SUBTITLE

Guide to Design, Implementation and Management of Distributed Databases

5. AUTHOR(S)

Elizabeth N. Fong; Charles L. Sheppard; Kathryn A. Harvill

6. PERFORMING ORGANIZATION (IF JOINT OR OTHER THAN NIST, SEE INSTRUCTIONS) U.S. DEPARTMENT OF COMMERCE NATIONAL INSTITUTE OF STANDARDS AND TECHNOLOGY GAITHERSBURG, MD 20899	7. CONTRACT/GRANT NUMBER
	8. TYPE OF REPORT AND PERIOD COVERED Final

9. SPONSORING ORGANIZATION NAME AND COMPLETE ADDRESS (STREET, CITY, STATE, ZIP)

Same as item #6

10. SUPPLEMENTARY NOTES

11. ABSTRACT (A 200-WORD OR LESS FACTUAL SUMMARY OF MOST SIGNIFICANT INFORMATION. IF DOCUMENT INCLUDES A SIGNIFICANT BIBLIOGRAPHY OR LITERATURE SURVEY, MENTION IT HERE.)

For an organization to operate in a distributed database environment, there are two related but distinct tasks that must be accomplished. First, the distributed database environment must be established. Then, a distributed database application can be designed and installed within the environment. This guide describes both of these activities based on a development life-cycle phase framework. This guide provides practical information and identifies skills needed for systems designers, application developers, database and data administrators who are interested in the effective planning, design, installation, and support for a distributed database environment. In addition, this guide instructs system analysts and application developers with a step-by-step procedure for the design, implementation and management of a distributed DBMS application.

This guide also notes that truly heterogeneous distributed database technology is still a research consideration. Commercial products are making progress in this direction, but usually with many update and concurrency restrictions and often with severe performance penalties.

The scope of this guide is based on experiences gained in the development of a distributed DBMS environment using two off-the-shelf homogeneous distributed database management systems. In support of the successful installation of the distributed database environment, a demonstration distributed application was designed and installed.

12. KEY WORDS (6 TO 12 ENTRIES; ALPHABETICAL ORDER; CAPITALIZE ONLY PROPER NAMES; AND SEPARATE KEY WORDS BY SEMICOLONS)

databases; DDBMS; distributed application; distributed database management systems; distributed environment; life-cycle phases.

13. AVAILABILITY		14. NUMBER OF PRINTED PAGES 59
XX	UNLIMITED	
	FOR OFFICIAL DISTRIBUTION. DO NOT RELEASE TO NATIONAL TECHNICAL INFORMATION SERVICE (NTIS).	
XX	ORDER FROM SUPERINTENDENT OF DOCUMENTS, U.S. GOVERNMENT PRINTING OFFICE, WASHINGTON, DC 20402.	15. PRICE
XX	ORDER FROM NATIONAL TECHNICAL INFORMATION SERVICE (NTIS), SPRINGFIELD, VA 22161.	

ELECTRONIC FORM * U.S. G.P.O.:1991-281-557:40224

ANNOUNCEMENT OF NEW PUBLICATIONS ON COMPUTER SYSTEMS TECHNOLOGY

Superintendent of Documents
Government Printing Office
Washington, DC 20402

Dear Sir:

Please add my name to the announcement list of new publications to be issued in the series: National Institute of Standards and Technology Special Publication 500-.

Name ____

Company ____

Address ____

City ______________ State ____________ Zip Code __________

(Notification key N-503)